МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Линейные структуры данных

Студент гр. 8304 _____ Карабанов Р.Е

Преподаватель _____ Фирсов М.А

Санкт-Петербург

2019

Цель работы.

Получить опыт работы с линейными структурами данных, изучит на практике реализацию.

Задание.

Вариант № 6 - В.

Проверить, является ли содержимое заданного текстового файла F правильной записью формулы следующего вида:

< формула > ::= < терм > | < терм > + < формула > | < терм > - < формула >

< терм > ::= < имя > | (< формула >) | [< формула >] | { < формула > }

< имя > ::= x | y | z

Описание функций краткое описание алгоритма.

Для проверки формулы использовался стек, реализованный с помощью классов, class stack, в котором были реализованны методы для работы со стеком. В начале работы пользователю предлагается выбрать способ ввода иерархического списка — с консоли или из файла. Проверка осуществляется с помощью метода execute(std::string str, unsigned int *i, char brackets = '0') который принимает указатель на индекс элемента в строке, введённой пользователем, указатель был выбран, во избежание глобальной переменной для отслеживания позиции символа в строке и переменную brackets для отслеживания рекурсии, в случае глубины рекурсии 0 brackets = «0». В методе проводится посимвольная проверка строки, в случае встречи <имя> или открывающая скобка производится добавления данного элемента в стек, в случае +/- удаление элемента из стека, в случае закрывающей скобки

проверка скобки на парность и удаление из стека. Далее с помощью метода isEmpty() проверятся пустота стека, формула верна, если стек пуст.

```
> Choose your input
> 0 - from console
> 1 - from file
> Any other to Exit
> 1
> FilePath: test
> Reading from file:

(z+z)
test #1 "(z+z)"
> Correct exp
(z-z-z)
test #2 "(z-z-z)"
> Correct exp
z+z-z+({z})-z
test #3 "z+z-z+({z})-z"
> Correct exp
z
test #4 "z"
> Correct exp
zz
test #5 "zz"
> Wrong exp
z+z
test #6 "z+z"
> Correct exp
z+x+y-(z-(zz))
test #7 "z+x+y-(z-(zz))"
> Wrong exp
z(z)
test #8 "z(z)"
> Wrong exp
```

Тестирование.

Ниже представлена таблица № 1, в которой представлены некоторые примеры работы программы.

Таблица №1 примеры работы программы.

Формула	Правильность
(z-z-(z)+x-y+((z-z)-(z))+x)	+
( )	-
(x}	-

Выводы.

В ходе лабораторной работы был получен опыт работы с линейными структурами данных, реализации одной из структур для определения правильность формулы.

Исходный код.

```
#include <iostream>
#include <string>
#include <fstream>

//< формула > ::= < терм > | < терм > + < формула > | < терм > - < формула >
//< терм > ::= < имя > | ( < формула > ) | [ < формула > ] | { < формула > }
//< имя > ::= x | y | z

class stack {
private:
   char *ptrStack;
   int top;
   int maxSize;
   inline char getTop();
   inline void expError();
   inline void resize();
   inline void push(char value);
   inline char pop();
public:
   inline stack(int = 20);
   inline ~stack();
   void execute(std::string, unsigned int *i, char brackets);
   inline void isEmpty();
   void init(){
      top = 0;
   }
};

inline stack::stack(int size) {
   maxSize = size;
   top = 0;
   ptrStack = new char[size];
```

```
}

inline stack::~stack() {
   delete[] ptrStack;
}

inline void stack::expError() {
   std::cout << "> Wrong exp\n";
}

inline char stack::getTop() {
        if (top != 0)
                return ptrStack[top - 1];
        else {
      expError();
      return 0; // Без этого warning
        }
}

inline char stack::pop() {
   if (top < 0){
      std::sout << «> Wrong exp\n»;
      exit(1);
   }
   top--;
   return ptrStack[top];
}

inline void stack::push(char value) {
   if (top == maxSize)
      resize();
   ptrStack[top] = value;
   top++;
}

inline void stack::resize() {
   ptrStack = (char*)realloc(ptrStack, maxSize * 2);
   maxSize *= 2;
}
```

```
inline void stack::isEmpty() {
   if (top == 0){
      std::cout << "> Correct exp\n";
      return;
   }
   expError();
}

bool pairBrackets(char openBrackets, char closedBrackets) {
   if ((openBrackets == '(' && closedBrackets == ')') || (openBrackets == '[' && closedBrackets ==
']') || (openBrackets == '{' && closedBrackets == '}'))
      return true;
   return false;
}

void stack::execute(std::string str, unsigned int *i, char brackets = '0') {
   for (; *i < str.length(); (*i)++){
      if (str.at(*i) == 'x' || str.at(*i) == 'y' || str.at(*i) == 'z') {
         push(str.at(*i));
      }
      else if (str.at(*i) == '(' || str.at(*i) == '[' || str.at(*i) == '{') {
         push(str.at(*i));
         (*i)++;
         execute(str, i, str.at(*i));
      }
      else if (str.at(*i) == '+' || str.at(*i) == '-') {
         pop();
      }
      else if (str.at(*i) == ')' || str.at(*i) == ']' || str.at(*i) == '}') {
         pop();
         if (brackets != '0') {
                                          if (pairBrackets(getTop(), str.at(*i)))
                                                   return;
            else{
               std::cout << "> Wrong exp\n";
               delete[] ptrStack; // непарные скобки, завершается работа.
               exit(1);
            }
```

```
                                }
        }
    }
    if (brackets == '0')
        pop();
}

void readFromFile(std::string filename, unsigned int *i, stack &s) {
            std::ifstream file(filename);
            if (file.is_open())
            {
                        std::cout << "> Reading from file:" << "\n\n";
                        int count = 0;
                        std::string exp;
                        while (std::getline(file, exp))
                        {
          s.init();
          (*i) = 0;
                                    count++;
                                    std::cout << exp << std::endl;
          std::cout << "test #" << count << " \"" + exp + "\"" << "\n";
                                    s.execute(exp, i);
          s.isEmpty();
                        }
            }
            else
            {
                        std::cout << "File not opened" << "\n";
            }
}

int main(int argc, char* argv[]) {
    unsigned int i = 0;
    stack s;
            std::cout << "> Choose your input" << std::endl;
            std::cout << "> 0 - from console" << std::endl;
            std::cout << "> 1 - from file" << std::endl;
            std::cout << "> Any other to Exit" << std::endl;
            std::cout << "> ";
```

```
        char command = '3';
        std::cin.get(command);
        std::cin.get();

        switch (command) {
        case '0': {
                std::string input;
                std::cout << "> Enter the string : ";
                std::getline(std::cin, input);
                s.execute(input, &i);
    s.isEmpty();
                break;
        }
        case '1': {
                std::cout << "> FilePath: ";
                std::string filePath;
                if (argc > 1) {
                        filePath = argv[1];
                        std::cout << filePath << std::endl;
                }
                else
                        std::cin >> filePath;
                readFromFile(filePath, &i, s);
                break;

        }
        case '3':
        default:
                std::cout << "> Error command \n> end\n";
        }
        return 0;
}
```